(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号

特開平9－319659

(43)公開日　平成9年（1997）12月12日

(51)Int.Cl.[6]	識別記号	庁内整理番号	ＦＩ		技術表示箇所
Ｇ０６Ｆ　12/14	３１０		Ｇ０６Ｆ　12/14	３１０	Ｋ
9/06	５５０		9/06	５５０	Ｚ

審査請求　未請求　請求項の数11　ＯＬ　外国語出願　有（全 42 頁）

(21)出願番号　特願平8－270111

(22)出願日　平成8年（1996）10月11日

(31)優先権主張番号　５４１６３６

(32)優先日　1995年10月10日

(33)優先権主張国　米国（ＵＳ）

(71)出願人　591244823
データー・ゼネラル・コーポレーション
ＤＡＴＡ　ＧＥＮＥＲＡＬ　ＣＯＲＰＯＲＡＴＩＯＮ
アメリカ合衆国マサチューセッツ州01581，ウエストボロー，ルート　9

(72)発明者　ケネス・ジョン・ヘイマン
アメリカ合衆国ノース・カロライナ州27713，ダーラム，フォーチュネス・リッジ・ドライブ　5500，アパートメント　79シー

(74)代理人　弁理士　湯浅　恭三　（外５名）

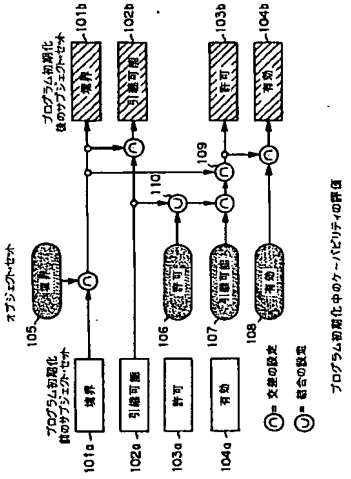
最終頁に続く

(54)【発明の名称】コンピュータ・システム用のセキュリティ制御方法

(57)【要約】

【課題】　コンピュータ・システム上の機能およびデータに対するアクセスに制限を賦課するコンピュータ・システム用のセキュリティ制御方法を提供する。

【解決手段】　コンピュータ機能の全体が、関連のケーパビリティ・セット１０１～１０８を有するイベントのセットに細分化され、ユーザがコンピュータ・システム上で行おうとする特定のジョブに応じて、各ユーザには、異なるケーパビリティが割り当てられる。各データ・ファイル、他のシステム資源、各プロセス上にはセキュリティ・ラベルが配置される。各プロセスにはある範囲の階層／カテゴリー・ラベル（ＭＡＣラベル）が割り当てられ、特別なケーパビリティを適用することができる副格子を定義する。



1

【特許請求の範囲】

【請求項1】　コンピュータ・システム用のセキュリティ制御方法であって、

（a）前記コンピュータ・システムのデータベースの中、および前記コンピュータ・システム自体の中に、セキュリティ・ポリシーを記憶するステップと、

（b）前記データベースおよび前記コンピュータ・システムにこれらの使用中に連続的にアクセスし、前記コンピュータ・システムを現在使用中の特定のサブジェクトに対して、前記データベースおよび前記コンピュータ・システムの中に記憶されている特定のポリシーを決定するステップと、

（c）前記ステップ（b）の結果によって許される範囲でのみ、前記サブジェクトに前記コンピュータ・システムへのアクセスを許可するステップと、を備え、

前記データベースおよびコンピュータ・システムは、各サブジェクトおよび各オブジェクトに対してマンダトリ・アクセス制御ラベルを含み、前記ラベルは階層格子に配列され、より高いレベルのラベルが管理領域に割り当てられ、中間レベルのラベルがユーザ領域に割り当てられ、より低いレベルのラベルがウイルス予防領域に割り当てられ、

更に、前記データベース内および前記コンピュータ・システム内に記憶されているセキュリティ・ポリシーに従って、1つのマンダトリ・アクセス制御領域内で動作中のサブジェクトには、該サブジェクトに特別なケーパビリティあるいはマンダトリ・アクセス制御属性の特別のセットが割り当てられなければ、他の領域内のオブジェクトにアクセスすることが許可されない、ことを特徴とする方法。

【請求項2】　請求項1記載の方法において、更に、前記データベースに記憶され、かつ前記コンピュータ・システム内に実施された前記セキュリティ・ポリシーに従って、サブジェクトには、前記ウイルス予防領域内のマンダトリ・アクセス制御ラベルを有するプログラムの実行のみが許可されていることを特徴とする方法。

【請求項3】　コンピュータ・システム用のセキュリティ制御方法であって、

（a）前記コンピュータ・システムのデータベースの中、および前記コンピュータ・システム自体の中に、セキュリティ・ポリシーを記憶するステップと、

（b）前記データベースおよび前記コンピュータ・システムにこれらの使用中に連続的にアクセスし、前記コンピュータ・システムを現在使用中の特定のサブジェクトに対して、前記データベースおよび前記コンピュータ・システムの中に記憶されている特定のポリシーを決定するステップと、

（c）前記ステップ（b）の結果によって許される範囲でのみ、前記サブジェクトに前記コンピュータ・システムへのアクセスを許可するステップと、を備え、

2

前記データベースおよびコンピュータ・システムは、各サブジェクトおよび各オブジェクトに対してマンダトリ・アクセス制御ラベルを含み、前記ラベルは階層格子に配列されており、

更に、前記データベースおよび前記コンピュータ・システム内に記憶されているセキュリティ・ポリシーに従って、マンダトリ・アクセス制御ラベルの範囲がサブジェクトに割り当てられ、該サブジェクトは、前記範囲内のマンダトリ・アクセス制御ラベルを有するオブジェクトへのアクセスを有する、ことを特徴とする方法。

【請求項4】　請求項3記載の方法において、前記データベースおよびコンピュータ・システムに記憶されている前記セキュリティ・ポリシーに従って、ある範囲のマンダトリ・アクセス制御ラベルが割り当てられたサブジェクトには、更に、前記範囲内のマンダトリ・アクセス制御ラベルを有するオブジェクトへの追加のアクセスを許す1組のケーパビリティが割り当てられることを特徴とする方法。

【請求項5】　コンピュータ・システム用のセキュリティ制御方法であって、

（a）前記コンピュータ・システムのデータベースの中、および前記コンピュータ・システム自体の中に、セキュリティ・ポリシーを記憶するステップと、

（b）前記データベースおよび前記コンピュータ・システムにこれらの使用中に連続的にアクセスし、前記コンピュータ・システムを現在使用中の特定のサブジェクトに対して、前記データベースおよび前記コンピュータ・システムの中に記憶されている特定のポリシーを決定するステップと、

（c）前記ステップ（b）の結果によって許される範囲でのみ、前記サブジェクトに前記コンピュータ・システムへのアクセスを許可するステップと、を備え、

前記データベースおよびコンピュータ・システムは、各サブジェクトおよび各オブジェクトに対してマンダトリ・アクセス制御ラベルを含み、前記ラベルは階層格子に配列されており、

更に、前記データベースおよび前記コンピュータ・システム内に記憶されているセキュリティ・ポリシーに従って、マンダトリ・アクセス制御ラベルの範囲がオブジェクトに割り当てられ、該オブジェクトは、前記範囲内のマンダトリ・アクセス制御ラベルを有するいずれかのサブジェクトにアクセス可能である、ことを特徴とする方法。

【請求項6】　コンピュータ・システム用のセキュリティ制御方法であって、

（a）前記コンピュータ・システムのデータベースの中、および前記コンピュータ・システム自体の中に、セキュリティ・ポリシーを記憶するステップと、

（b）前記データベースおよび前記コンピュータ・システムにこれらの使用中に連続的にアクセスし、前記コン

ピュータ・システムを現在使用中の特定のサブジェクトに対して、前記データベースおよび前記コンピュータ・システムの中に記憶されている特定のポリシーを決定するステップと、

（ｃ）前記ステップ（ｂ）の結果によって許される範囲でのみ、前記サブジェクトに前記コンピュータ・システムへのアクセスを許可するステップと、を備え、

前記データベースおよびコンピュータ・システムは、各サブジェクトおよび各オブジェクトに対してマンダトリ・アクセス制御ラベルを含み、前記ラベルは階層格子に配列され、各ラベルは二進セキュリティ属性を有し、

更に、前記データベースは、各マンダトリ・アクセス制御ラベルに対してテキスト属性を含み、該テキスト属性は、各ラベルに対する前記それぞれの二進セキュリティ属性と共に記憶されている、ことを特徴とする方法。

【請求項７】　コンピュータ・システム用のセキュリティ制御方法であって、

（ａ）前記コンピュータ・システムのデータベースの中、および前記コンピュータ・システム自体の中に、セキュリティ・ポリシーを記憶するステップと、

（ｂ）前記データベースおよび前記コンピュータ・システムにこれらの使用中に連続的にアクセスし、前記コンピュータ・システムを現在使用中の特定のサブジェクトに対して、前記データベースおよび前記コンピュータ・システムの中に記憶されている特定のポリシーを決定するステップと、

（ｃ）前記ステップ（ｂ）の結果によって許される範囲でのみ、前記サブジェクトに前記コンピュータ・システムへのアクセスを許可するステップと、を備え、

前記データベースおよび前記コンピュータ・システムは、オブジェクトに対する複数のケーパビリティを含む要求ケーパビリティ・セットと、各サブジェクトに対する複数のケーパビリティを含む有効ケーパビリティ・セットとを含み、

更に、前記データベース内に記憶されているセキュリティ・ポリシーに従って、サブジェクトの有効ケーパビリティ・セットが前記オブジェクトの要求ケーパビリティ・セット内のケーパビリティ全てを含む場合にのみ、前記サブジェクトは前記オブジェクトにアクセスすることが許される、ことを特徴とする方法。

【請求項８】　コンピュータ・システム用のセキュリティ制御方法であって、

（ａ）前記コンピュータ・システムのデータベースの中、および前記コンピュータ・システム自体の中に、セキュリティ・ポリシーを記憶するステップと、

（ｂ）前記データベースおよび前記コンピュータ・システムにこれらの使用中に連続的にアクセスし、前記コンピュータ・システムを現在使用中の特定のサブジェクトに対して、前記データベースおよび前記コンピュータ・システムの中に記憶されている特定のポリシーを決定するステップと、

（ｃ）前記ステップ（ｂ）の結果によって許される範囲でのみ、前記サブジェクトに前記コンピュータ・システムへのアクセスを許可するステップと、を備え、

前記データベースは、

サブジェクトが以降のプログラム実行に渡すことができる複数のケーパビリティを含むサブジェクト引継可能ケーパビリティ・セットと、

サブジェクトに割り当てられる全ケーパビリティについての絶対上限を含むサブジェクト境界ケーパビリティ・セットと、を含み、

更に、前記データベースおよび前記コンピュータ・システムに記憶されている前記セキュリティ・ポリシーに従って、サブジェクトがそのサブジェクト引継可能ケーパビリティ・セット内のケーパビリティに渡すことが、かかるケーパビリティが前記サブジェクトの境界ケーパビリティ・セット内に含まれていない場合、禁止されることを特徴とする方法。

【請求項９】　ユーザが特にコンピュータに実行を要求し且つユーザが通常実行するケーパビリティを有するユーザ・モード・オペレーションと、システムの保守および保全性の目的のためにシステムが実行するシステム・モード・オペレーションと、システムが数個の選択ケーパビリティを与え且つユーザによって与えられるそれらを強化する増補ユーザ・オペレーションとを含むプログラムをコンピュータ上で実行する方法であって、

ユーザ・モード・オペレーションの連続列が前記コンピュータによって実行されている時間中、サブジェクトの有効ケーパビリティ・セットをその引継可能ケーパビリティ・セットに等しく設定するステップと、

連続システム・モード・オペレーションの列が前記コンピュータによって実行されている時間中、サブジェクトの有効ケーパビリティ・セットをその許可ケーパビリティ・セットに等しく設定するステップと、

ユーザ・モード・オペレーションの後続の連続列が前記コンピュータによって実行されている時間中、サブジェクトの有効ケーパビリティ・セットをその引継可能ケーパビリティ・セットに等しく設定するステップと、

連続増補ユーザ・モード・オペレーションの列が前記コンピュータによって実行されている時間中、サブジェクトの有効ケーパビリティ・セットを、その引継可能ケーパビリティ・セットと、厳格に定義された増補するケーパビリティ・セットとを加えたものに等しく設定するステップと、を備えることを特徴とする方法。

【請求項１０】　コンピュータ・システム用のセキュリティ制御方法であって、

（ａ）前記コンピュータ・システムのデータベースの中、および前記コンピュータ・システム自体の中に、セキュリティ・ポリシーを記憶するステップと、

（ｂ）前記データベースおよび前記コンピュータ・シス

テムにこれらの使用中に連続的にアクセスし、前記コンピュータ・システムを現在使用中の特定のサブジェクトに対して、前記データベースおよび前記コンピュータ・システムの中に記憶されている特定のポリシーを決定するステップと、

（ｃ）前記ステップ（ｂ）の結果によって許される範囲でのみ、前記サブジェクトに前記コンピュータ・システムへのアクセスを許可するステップと、を備え、

前記データベースは、ユーザが前記コンピュータ・システムへの初期アクセスを許される前に正確に応答しなければならない複数の認証機構を含み、

更に、各ユーザには、該ユーザがネットワークへの初期アクセスを許される前に正確に応答しなければならない異なるセットの認証機構を割り当て可能である、ことを特徴とする方法。

【請求項１１】　コンピュータ・システム用のセキュリティ制御方法であって、

（ａ）前記コンピュータ・システムのデータベースの中、および前記コンピュータ・システム自体の中に、セキュリティ・ポリシーを記憶するステップと、

（ｂ）前記データベースおよび前記コンピュータ・システムにこれらの使用中に連続的にアクセスし、前記コンピュータ・システムを現在使用中の特定のサブジェクトに対して、前記データベースおよび前記コンピュータ・システムの中に記憶されている特定のポリシーを決定するステップと、

（ｃ）前記ステップ（ｂ）の結果によって許される範囲でのみ、前記サブジェクトに前記コンピュータ・システムへのアクセスを許可するステップと、を備え、

前記データベースは、複数のケーパビリティを含む管理ケーパビリティ・セットを含み、各ケーパビリティは、管理者がサブジェクトに授与することができる異なるタイプの特権であり、

更に、前記データベース内に記憶されているセキュリティ・ポリシーに従って、管理者は、サブジェクトを、当該管理者の管理ケーパビリティ・セット内にないケーパビリティに割り当てることを禁止されている、ことを特徴とする方法。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、コンピュータ・システム、特にコンピュータ・システム用の１組のセキュリティ機構に関するものである。このセキュリティ・システムは、誰がコンピュータ・システムに対してアクセスを有するかと、一旦システムがアクセスされたときシステムの資源へのアクセスの範囲とを正確に制御するものである。

【０００２】

【従来の技術】１つの会社または政府の機関内では、増々多くのコンピュータが共に接続されて大きなコンピュータ・ネットワークとなり、時として地球全体に延びているので、コンピュータ・システム上の情報を、無許可のアクセスから保護することが増々重要となりつつある。このような、無許可のアクセスは、組織の外側から来る可能性があるが、組織内部から来る場合も多い。例えば、低位従業員が、従業員の給与や組織の将来の計画に関する機密情報へのアクセスを得る可能性もある。

【０００３】本願の譲受人である、データ・ジェネラル・コーポレーション(Data GeneralCorporation)は、この分野では、以前から活動的であり、１９９４年８月にコンピュータ・セキュリティ・システムを導入した。このコンピュータ・セキュリティ・システムは、データ・ジェネラルの産業用の先進のＤＧ／ＵＸユニックスを基本とするオペレーティング・システムと密接に統合化され、ＤＧ／ＵＸオペレーティング・システム・アーキテクチャの一体的構成物を形成した。このセキュリティ・システムは、あるレベルの制御を提供するものではあったが、以下のような欠点に苦慮していた。

【０００４】従来のセキュリティ・システムに伴う主な問題の１つは、当該システム上でユーザに管理動作をさせるためには、このユーザにシステム全体への全アクセスを与えることが必要であった。例えば、低位の管理ユーザを大会社の管理部門に雇用して有効なユーザのログを維持する場合、この低位ユーザにはこのコンピュータ・システム全体への「超ユーザ（ｓｕｐｅｒ－ｕｓｅｒ）」アクセスを与えなければならず、従って個々の個人的ファイルにアクセスしようとすれば可能であり、更に会社の機密データを読み出したり、および／または改変する潜在的な可能性もある。他の問題は、コンピュータ・システムに侵入し得るウイルスに関するものである。従来のシステムは、貴重なファイルを保護するためにウイルスを隔離せず、また管理ユーザによる実行の場合でも最初の場所でウイルスが発生するのを防止するための積極的な対策を取っていなかった。

【０００５】

【発明が解決しようとする課題】本発明は、コンピュータ・システム用のセキュリティ・システムを対象とし、だれがコンピュータ・システム上の正確にどのコンピュータ機能およびデータに対するアクセスを有するかについて、特有な制限を賦課するものである。本発明のセキュリティ・システムの特定の実施形態の結果として、ウイルスが安全に含まれ、記憶されているプログラムまたはデータを破壊することができる領域にウイルスが広がるのを防止する。また、ウイルスが多数のインスタンスに侵入または伝搬することも防止する。

【０００６】

【課題を解決するための手段】本発明のセキュリティ・システムは、基本的に、コンピュータ機能の全体を、必要とされる特権(privilege)に分割し、ユーザがコンピュータ・システム上で行う特定のジョブに応じて、各ユ

ーザに異なる特権を割り当てることを含む。

【０００７】また、各データ・ファイルまたはその他のシステム資源上に、および各ユーザ・プロセス上に、セキュリティ・ラベルを配置する。ラベルには階層を設け、高度に秘密のものから共通にアクセス可能なものまでの範囲とし、これらのラベルに基づいて、セキュリティ・システムは厳しいポリシーを実施し、誰がどのタイプのアクセスをどのデータ・ファイルまたは他のシステム資源に対して有するのかを決定する。本発明によれば、これらのラベル範囲は、特定のユーザ・プロセスに割り当てられ、当該プロセスの動作が許される許可範囲(clearance range)を規定する。更に、ラベルの階層を少数（例えば３）の領域に分割し、１つの領域上で動作するユーザ・プロセスは、通常、非常に注意深く指定される場合を除いて、他の領域にアクセスすることを許されていない。

【０００８】更に、あるデータ・ファイルの所有者は当該ファイルに制限を設けることを許されており、そのためあるケーパビリティを所有するユーザのみが当該ファイルへのアクセスを獲得することができるようになっている。

【０００９】

【発明の実施の形態】

定義

本出願において共通して用いられている以下の用語を次のように定義する。

【００１０】サブジェクト：オブジェクトにアクセスするか、またはシステム状態を変化させるコンピュータ・システムのアクティブなエンティティ（例えば、ユーザの代わりに動作するプロセス）。

【００１１】プロセス：ユーザの代わりに多くの場合実行するアクティブに実行中のプログラムであって、そのユーザが誰であるのかを識別し、そのユーザのアクセス権によって具現化される１組の信用証明書(credential)を有する当該アクティブに実行中のプログラム。

【００１２】オブジェクト：オペレーションの目標であり、多くの場合、データを受信し且つ記憶するコンピュータ・システムの受動要素（例えば、データ・ファイルまたはプログラム）。

【００１３】オペレーション：システムからのサービスに対する要求（例えば、読出しおよび書込みアクセスのためにファイルを開く要求）。

【００１４】特権のためのケーパビリティ機構(capability mechanism)

本発明のセキュリティ・システムは、プロセスがその存在の間呼び出すことができる１組のケーパビリティ（または特権）を各プロセスに割り当てることができる。ケーパビリティの一例は、システムが通常であればアクセスを否定するファイルを開くアビリティ(ability)である。別の例は、システムを停止するような、通常のユーザには利用できない限定されたオペレーションを実行することである。プロセスが必要とするときに呼び出すことができる１組の可能なケーパビリティのことをサブジェクト許可ケーパビリティ・セット(subject permitted capability set)（図１の１０３ａおよび１０３ｂを参照）と呼ぶ。

【００１５】各プロセスは、サブジェクト有効ケーパビリティ・セット(subject effectivecapability set)（１０４ａおよび１０４ｂ）も有する。このセットは、許可ケーパビリティ・セットのサブセットであり、上記プロセスが現在アサート(assert)しオペレーションを実行する１組のケーパビリティのことである。即ち、有効セットは、システムがアクセス制御判断を行うときに考慮するものであり、一方、許可セットは有効セットに含まれ得る上限である。

【００１６】時として、ある特定のプログラムの実行中に、あるプロセスの許可ケーパビリティ・セットを一時的に拡張し、ユーザが通常の業務の間には所持することはないケーパビリティを含まなければならないことがある。例えば、ユーザが彼のパスワードの変更を決定する場合がある。これを行うためには、ユーザは彼のパスワードを記憶しているシステム・ファイル、および関連する情報を更新しなければならない。しかしながら、ユーザは通常これらのファイルへのアクセスを許可されていないので、パスワード変更プログラムが、これらのファイルにアクセスするアビリティを与えなければならない。このプログラムは、特別なケーパビリティがその意図に沿った目的のためにのみ用いられることを保証する任務を担い、上記特別のケーパビリティはパスワード変更プログラムが終了したときには自動的に除去される。

【００１７】このような特別なケーパビリティは、プログラム上では、オブジェクト許可ケーパビリティ・セット（１０６）として記憶される。このセットにおけるケーパビリティは、あるプロセスがプログラムを実行している間に、一時的にそのプロセスに認可される。オブジェクトはまた、それに関連する他のケーパビリティ・セットを有することができ、そのことは図１に示されるようにプログラムを実行するいずれかのプロセスに割り当てられるケーパビリティを変更する。例えば、あるプロセスがあるプログラムの実行を開始したとき、オブジェクト許可ケーパビリティは、セット結合演算子(set union operator)（１１０）によって、当該プロセスの許可ケーパビリティに付加され、サブジェクト許可セットは、セット交差演算子(set intersection operator)（１０９）によって、サブジェクト境界セット(subject bounding set)（以下で論ずる）に制限される。

【００１８】セット１０２ａは、サブジェクト引継可能ケーパビリティ・セット(subject inheritable capability set)であり、あるプロセスがそのプロセスによって実行された後のプログラムに渡すことができるケーパビ

リティのセットを表わす。即ち、このセットにおけるケーパビリティは、いずれのオブジェクト許可ケーパビリティ・セットからの付加もなく、このプロセスが通常所有するケーパビリティのことである。プログラムの初期化以前のサブジェクト引継可能セット（１０２ａ）は、プログラムの開始後は新たなサブジェクト引継可能セット（１０２ｂ）となるが、本出願人の境界セットに関する追加セキュリティ対策を除く。これについては以下で説明する。オブジェクト許可ケーパビリティ・セットから獲得したケーパビリティを、後のプログラムに渡すのを防止するのは、この機構によるものである。何故なら、このような強化されたケーパビリティはサブジェクト引継可能セットに付加されることは決してないからである。

【００１９】上述の構造は、過去においてＩＥＥＥ　ＰＯＳＩＸセキュリティ・システムに含まれていた。本出願人は、上述の基本構造にフェールセーフ機構を付加し、次のようなものとした。

【００２０】サブジェクト境界ケーパビリティ・セット(subject bounding capability set)（１０１ａ、１０１ｂ）を定義する。これは、プロセスが獲得し得るケーパビリティの最大のグループである。プロセスはその境界セットを増加させることは不可能である。図１に示すように、このセットは、減少させることができ（例えば、あるオブジェクト境界セットが、元のサブジェクト境界セットよりも小さい場合）、他のサブジェクト・ケーパビリティ・セット全てに対する制限として役立つ。これにより、ケーパビリティを獲得するプロセスのアビリティを効果的に低減するので、フェールセーフ機構として役立つ。例えば、管理機能を実行することを許されていないユーザには、比較的小さい境界セットを割り当てればよい。たとえこのユーザが管理の識別(administrative identity)を引き受ける(takeon)よう管理するとしても、その境界セットは、それらが管理者に関連するケーパビリティを獲得するのを防止し、従って、システムに損傷を与える潜在性を大幅に低減する（または排除すらする）ことになる。

【００２１】更に別のチェックとして、各実行可能プログラムは、オブジェクト境界ケーパビリティ・セット（１０５）も有することができ、当該オブジェクト境界ケーパビリティ・セットはプログラムの初期化中にサブジェクト境界ケーパビリティ・セット（１０１ａ）と交差(intersect)させて、プログラムが実行している間有効となる新たなサブジェクト境界セット（１０１ｂ）を形成する。この新たな境界セットはまた他のサブジェクト・ケーパビリティ・セットの全て、特に許可セット（１０３ｂ）および有効セット（１０４ｂ）を制限するので、これは所与のプログラムが特定のケーパビリティで実行されるのを防止する機構を提供する。

【００２２】<u>イベント機構</u>

インプリメンテーションが、特権をチェックすることを望むときは常にプロセスのサブジェクト有効セット内の特定のケーパビリティの存在を直接チェックするならば、ケーパビリティに関して実施される実際のセキュリティ・ポリシーは、本出願人によって決定されなければならず、個々のサイトはその同一ポリシーを使用しなければならないであろう。これを避け、システムによって実施されるセキュリティ・ポリシーを各サイトで構成可能とすることによって個々の要件を満足するようにするためには、特権チェックを「イベント」によって行う。イベントとは、「セキュリティに関連する判断がなされるまたは記録される、コードの中の位置」として定義する。あるオペレーションをプロセスに実行させるべきか否かをチェックする必要がシステムに生じたとき、そのシステムはその特定の特権チェックに定義された唯一のイベント名を用いて、当該プロセスが所有しチェックに渡さなければならない１つまたは複数のケーパビリティを表の中で検索する。個々のサイトに、イベント名対ケーパビリティの表を修正する機構を設けることによって、サイトは、セキュリティ・システムと共に送出され供給されたイベント／ケーパビリティ・ポリシーに加えて、またはこれに代えて、それら自身のセキュリティ・ポリシーを実施するようシステムを形成することが可能となる。

【００２３】<u>オペレーションのブラケット化(bracketing)</u>

サブジェクトがあるオブジェクト・プログラムを実行している間、プロセスのサブジェクト有効ケーパビリティ・セット（１０４ｂ）の状態は周期的に変化し、現在実行中のオペレーションのタイプを反映する。３つの異なるタイプのオペレーションが定義される。

【００２４】第１に、ユーザ・オペレーションは、ユーザに通常割り当てられているケーパビリティのみを用いて実行すべきものである。例えば、ユーザがファイルの印刷を要求したとき、要求されたファイルにアクセスするのは、ユーザ・オペレーションである。この場合、プロセスの有効ケーパビリティ・セット（１０４ｂ）は、その引継可能セット（１０２ｂ）に等しく設定される。なぜなら、これはオブジェクト許可セット（１０６）からの寄与がない、ユーザのケーパビリティを含むからである。この状態は、プログラムがユーザ・オペレーションを実行している限り維持される。

【００２５】第２に、システム・オペレーションは、可能であればシステムが引き継がなければならないものである。例えば、先の印刷要求において、システムは、それを印刷する前に、通常ユーザにはアクセス不可能なスプール・エリアにファイルをコピーする場合がある。このコピーは、ファイルの印刷を引き継がなければならないので、システム・オペレーションである。この場合、プロセスの有効ケーパビリティ・セット（１０４ｂ）

は、その許可ケーパビリティ・セット（１０３ｂ）に等しく設定される。この許可ケーパビリティ・セットは、ユーザのケーパビリティにいずれかのオブジェクト許可ケーパビリティ（１０６）を加えたものを表わす。

【００２６】最後に、増補(augmented)ユーザ・オペレーションは、本質的にはユーザ・オペレーションであるにも拘らず、オブジェクト許可セット（１０６）からの１つ以上のケーパビリティへのアクセスを必要とするものである。一例は、あるオブジェクト許可ケーパビリティ・セットをプログラム上で設定する場合である。これは、ユーザや管理者が通常所有できない特殊なケーパビリティを必要とする。このようなケーパビリティ・セットを設定するためのプログラムがオペレーションを許可する決定をした場合、この特定のケーパビリティをイネーブルする必要があるが、それが有し得る他のオブジェクト許可ケーパビリティをイネーブルすべきではない。なぜなら、これは、ユーザが通常有することはない目標プログラムへのユーザ・アクセスを認可し得るからである。この場合、プロセス有効セット（１０４ｂ）はサブジェクト引継可能セット（１０２ｂ）に等しく設定され、サブジェクト許可セット（１０３ｂ）からの特定のケーパビリティが付加される。

【００２７】「ブラケット化」という用語は、システム・オペレーションまたは増補ユーザ・オペレーションの丁度前にサブジェクト有効セット（１０４ｂ）内の特別なケーパビリティをイネーブルし、そのオペレーションの直後にそれらをディスエーブルすることを意味する。従来技術では、ブラケット化は、各ユーザ・オペレーションの付近で行われていた（列状に沢山あったとしても）。これは、各オペレーションに必要なケーパビリティを決定するための多大な分析と、更に、有効セット（１０４ｂ）をかなり頻繁に計算し修正するための計算上の努力とを必要とした。

【００２８】本発明によれば、図２に示すように、プログラムは、開始時に、ユーザ・オペレーションのために有効なケーパビリティ状態（１０４ｂ）を確立する。このケーパビリティ状態は、プログラムがユーザ・オペレーションを実行している限り保持される。システム・オペレーションまたは増補ユーザ・オペレーションを実行する時刻が来たとき、そのオペレーションの直前に有効ケーパビリティ・セット（１０４ｂ）の中で適切なケーパビリティをイネーブルし、そのオペレーションの直後に有効ケーパビリティ・セット（１０４ｂ）の中でユーザ・オペレーション・ケーパビリティ・セットをリセットする。システム・オペレーションまたは増補ユーザ・オペレーションの前後に特別のケーパビリティをイネーブルおよびディスエーブルすることにより、オペレーションのまわりに「ブラケット」を形成する。このようにして、特別なケーパビリティをイネーブルするときに、システム・オペレーションまたは増補ユーザ・オペレーションを識別する特定の場所を除いて、プログラムは通常ユーザのケーパビリティのみで実行する。あるシステム・オペレーションに特定して必要なケーパビリティのみの代わりに、当該システム・オペレーションに使用可能な全てのケーパビリティをイネーブルすることは受容可能である。それは、全てのオペレーションは、余分な無関係のケーパビリティがシステム・オペレーションの成果には影響を与えないようにしてあるからである。

【００２９】本発明は、従来技術よりもブラケット化を行う頻度はかなり少ないので、多量の分析および計算努力が節約される。即ち、各ユーザ・オペレーションに対して有効ケーパビリティ・セット（１０４ｂ）を調節する必要がない。代わりに、ユーザが通常実行する場合に用いられるケーパビリティが用いられる。

【００３０】マンダトリ・アクセス制御(mandatory access control)

上述のケーパビリティの検討は、あるプロセスが正確にどのオペレーションを実行できるかに関して、そのプロセスを抑制することができるという、セキュリティ・システムのアビリティに関するものであった。本発明によれば、特定のオブジェクトへのサブジェクトのアクセスを更に抑制することが可能である。

【００３１】周知のマンダトリ・アクセス制御（ＭＡＣ）理論によれば、各オブジェクトおよび各サブジェクトにラベルを付与する。各ラベルは次の２要素で構成される。

【００３２】１．階層要素。これは、順番に並べられ、感度レベル（例えば、無分類、秘密、極秘等）を示す。

【００３３】２．カテゴリー・セット要素。順番には並べられず、対象となるエリア(area)（単数または複数）を示す（例えば、マーケティング、研究および開発、人事等）。

【００３４】これらのＭＡＣラベルは数学的な格子を形成する。このようなラベルの一例は、「秘密：（スーパーカー、マーケティング）」である。このラベルは、感度レベルが「秘密」であり、対象エリアが「スーパーカー」および「マーケティング」エリアの双方に関するオブジェクト（例えば、マーケット調査情報を含むデータファイル）に関する。

【００３５】１．第１のラベルの階層要素が第２のラベルの階層要素より高いか又はそれに等しく、そして
２．第２のラベルのカテゴリー・セットが第１のラベルのカテゴリー・セットのサブセット（即ち、これに含まれる）の場合、第１のラベルは第２のラベルを支配する（即ち、第２のラベルよりも高い）。

【００３６】従って、２つのラベルが与えられると、第１が第２を支配する場合や、各々が他方を支配する（この場合、２つのラベルが等しい）場合、あるいはいずれも他方を支配しない（この場合、ラベルは「比較不可」と言われる）場合がある。

【００３７】サブジェクトは、当該サブジェクトのラベルが他のオブジェクトのラベルを支配するとき（ラベルが等しい場合を含む）、このオブジェクトを読むことができる。これは「リードーダウン(read-down)」と呼ばれる。サブジェクトは、オブジェクト・ラベルがこのサブジェクト・ラベルを支配する場合、このオブジェクトに書き込むことができる。これは「ライトーアップ(write-up)」と呼ばれる。

【００３８】本発明は、「リードーダウン、ライトーイコール(write-equal)」の基本ポリシーをより厳格に実施するものであり、サブジェクトがオブジェクトに書き込みができるのは、サブジェクト及びオブジェクトのラベルが等しいときのみである。

【００３９】MAC範囲(ranges)

従来技術では、特定のMACラベルにおいて実行中のプロセスは、そのプロセスのラベルによって支配されているラベルを有するオブジェクトのみを読み取ることができるに過ぎなかった。この許されている基本ポリシー以上のアクセスをユーザに与えたいならば、ユーザのプロセスに、コンピュータ・システム内の全オブジェクトへの全体的なアクセスを認可しなければならないであろう。ＵＮＩＸオペレーティング・システムでは、このようなユーザは、「スーパーユーザ」（または「ルート(root)」）として知られている。１つの制限されたオペレーションを行うためには、制限されたオペレーション全てを実行するアビリティをユーザに与えなければならない。通常、これは実用上全く望ましくないことである。

【００４０】本出願人は、上述のケーパビリティ機構の組み合わせと、プロセスにMAC範囲（MACラベルの範囲）を割り当てることによって（プロセスのMACラベルに加えて）、この問題を克服した。図３に示すように、この範囲は上側境界（範囲の中で最も高いMACラベル）と、下側境界（範囲の中で最も低いMACラベル）とを有し、上側境界が下側境界を支配する。このようなMAC範囲が、MAC格子全体の副格子を形成する。

【００４１】ケーパビリティ機構は、あるプロセスが当該プロセスのMAC範囲以内に在るラベルを有するオブジェクトにアクセスするときに、基本的なシステムのMACポリシーをオーバライドするのを可能にするケーパビリティを与える。例えば、あるプロセスのケーパビリティは、MACラベルがプロセスのラベルよりも高いが、尚そのプロセスのMAC範囲内にある（「範囲内ライトーアップ(write-up-within-range)」）いずれかのオブジェクトに、書込みアクセスを許可するように構成することができる。他のケーパビリティは、「範囲内リードーアップ(read-up-within-range)」および「範囲内ライトーダウン(write-down-within-range)」を許可する。これらは、上述と同様の定義を有する。

【００４２】本出願人のMAC範囲は、ケーパビリティ機構と結合されているので、厳格に定義されたオブジェクト・セット（即ち、プロセスのMAC範囲内のMACラベルを有するもの）については、ユーザへのアクセス付与を増やし、一方、システム上の他のオブジェクトについては、同一ユーザへのアクセス付与を増やさないようにすることが可能である。

【００４３】上述の実施例（図４参照）に加えて、MAC範囲を特定のオブジェクトに関して定義することも可能である。サブジェクトのMACラベルがオブジェクトのMAC範囲の上側および下側境界の間にある限り（図４のサブジェクト１、２および３の各々）、当該オブジェクトへの書き込みのみのアクセスが認可される。読出しアクセスは、サブジェクトのMACラベルがオブジェクトのMAC範囲の上側境界を支配することを必要とする。従って、読出し／書込みアクセスは、MACラベルがオブジェクト範囲の上側境界に等しいプロセスのみに認可される。

【００４４】このように、種々のMACラベルにおけるある範囲のユーザには、他の者が書き込んだものを読み出すアビリティを必ずしも彼らに与えることなく、オブジェクト（例えば、ユーザが日々のトランザクションを記録するログ・ファイル）への書き込みを許可することができる。

【００４５】MAC領域(regions)

基本的なMACポリシー（リードーダウン、ライトーイコール）は限定的であり、その結果種々の問題が生じることを本出願人は発見した。

【００４６】第１に、管理者が管理作業を行っている場合、当該管理者は彼らのプロセスのラベルより低いあらゆるMACラベルへの「ライト・ダウン」を行うことができる。多くの管理プロセスは、典型的に、MAC格子の上側部分で実行され、そのため通常の（非管理）ユーザは管理オブジェクトにアクセスすることができない。従って、低レベルのユーザ（例えば、高校生）に対して限られた管理的役割を与える場合（各月毎に多くの従業員を得たり失ったりする大会社における、ユーザの追加および削除のようなもの）、そのユーザはそれらの管理MACラベルによって支配されているものであれば何にでもアクセスすることができよう。一例として、低レベル管理従業員は、上級会社役員(senior company officials)の電子メールボックスへのアクセスを得ようとすれば、得ることができよう。

【００４７】第２に、あるユーザによって実行されるプログラムにウイルスのような悪意のコードが含まれていることもあり、これによって、システム上の他のプログラムやファイルが変更されたり破壊される可能性がある。管理者がこのようなプログラムを実行すると、システムの実行可能なもの(system executables)が変更され、他のユーザによってその後実行されたときに、秘密保持を意図している情報を開示してしまうというような

無許可の動作を行う可能性がある。

【００４８】本出願人は、ＭＡＣ階層全体を、少数の別個の重複しない領域に分割することによって、これらの問題を解決した。例えば、図５に示すように、３つの領域を使用することができる。即ち、管理領域（４１）、ユーザ領域（４２）およびウイルス防止領域（４３）である。管理領域は、最も高いＭＡＣ階層に位置付けられ、ウイルス防止領域は最も低いＭＡＣ階層にあり、ユーザ領域はその中間にある。有効なポリシーの制約が他の領域内のオブジェクトにアクセスする１つの領域内のプロセスに置かれており、そのため上述のような問題が克服される。

【００４９】例えば、管理領域内のＭＡＣラベルで実行中のプロセスは、その（管理）ＭＡＣラベルではなく、（定義によりユーザ領域内にある）そのＭＡＣ範囲の上側境界を介して、ユーザおよびウイルス防止領域内のオブジェクトへのアクセスを得る。上限がユーザ領域の最下部にあるＭＡＣ範囲を管理者に与えることによって、その管理者がユーザ領域内のファイルの殆どまたは全てにアクセスするのを防止することが可能である。

【００５０】同様に、ウイルス防止領域に書き込みを行うには特殊なケーパビリティを必要とし、このケーパビリティは、通常、ソフトウエアのインストールを担う管理者にのみ認可され、他の管理者またはユーザには認可されない。システムの実行可能なものはウイルス防止領域に記憶される、そのためこれらはこの書込み保護の恩恵を受ける。プロセスがプログラムを実行するとき、このプロセスに割り当てたメモリの中にプログラムのコピーが作成される。プログラムが、ウイルス防止領域（全てのシステムの実行可能なものおよび多くの他のシステム・ファイルが配置されている）内のいずれかのオブジェクトを変更させようとする悪意のあるコードを含む場合、アクセスは否定される。従って、ウイルスは実効的に含まれ、拡散し損傷を広げることは不可能となる。

【００５１】プロセスが異なるＭＡＣ領域内のオブジェクトへのアクセスに対する制約をオーバライドすることを可能にするケーパビリティはない。これは、種々の管理者のアクセスを、そのジョブを遂行してもらうのに必要なオブジェクトのみに限定することによって、管理権限の抑制を図ったものである。

【００５２】トラステッド・ファシリティ・モード(Trusted Facility Mode)

先に論じたＭＡＣ領域という態様に加えて、本発明は、あるプログラムがウイルス防止領域（４３）内に記憶されている場合、プロセスにはそのプログラムの実行のみが許されるという点において、特別なセキュリティおよび特別なウイルス保護を提供する。これは、例えば、各プログラムを実行する前にそれに対するチェックを行い、ソフトウエアがウイルス防止領域（４３）内に存在することを確かめることによって達成可能である。従って、ウイルス防止領域（４３）は、トラステッド・ファシリティ・モードで実行するプロセスのために潜在的に実行可能なプログラムを含み得る唯一の場所である。これは、システム管理者はどのプログラムがユーザに使用可能となるかを正確に判断することができるので制御量を拡大させることになる。

【００５３】本発明のトラステッド・ファシリティ・モードという態様によって、ユーザは、例えば、インターネットからダウンロードしたり、プログラムを実行する（例えば、ファイル転送プロトコルを用いて）ことができなくなる。外部プログラムは共通のウイルス源である。従って、外部プログラムがコンピュータ・システム上で実行するのを防止することによって、多数のインスタンスにおいてウイルスを防止することができる。

【００５４】トラステッド・ファシリティ・モードは、管理者がウイルス防止領域に公式にインストールされていないプログラムを実行するのを防止することによって、管理的な乱用を防止するためにも役立つものである。

【００５５】マルチレベル・ディレクトリ(Multilevel Directories)

ファイル・システム・オブジェクトを作成するとき、サブジェクトが実際にディレクトリに書き込みを行う。ディレクトリは、それが収容する各ファイル・システム・オブジェクトをリストに纏めたものである。一般的なＭＡＣポリシーによれば、サブジェクトは、書き込むべきオブジェクトと同一のＭＡＣレベルでなければならない。この場合、オブジェクトはディレクトリであり、特定のＭＡＣレベルにある。あるＵＮＩＸディレクトリ（/temp, /var/mail等のような）は、多くのユーザがアクセスを有するべき、「共通の」ディレクトリである。従って、一般的なＭＡＣポリシーが適所にあるとき、ディレクトリのＭＡＣレベルにないサブジェクトがディレクトリに対して書き込むことは不可能である。

【００５６】本発明は、マルチレベル・ディレクトリを用いて、情報を異なるＭＡＣレベルのサブジェクトからディレクトリに受け入れるようにする。正しいＭＡＣレベルにおいて隠れサブディレクトリ(hidden subdirectory)が、書き込まれる情報に対して、動的に作成される。隠れディレクトリの各々が別個のＭＡＣレベルに存在し、隠れディレクトリがマルチレベル・ディレクトリ・ファミリに書き込みたい各サブジェクトＭＡＣレベルに対して作成される。サブジェクトがこの目的のために設計された特殊なケーパビリティを携えていなければ、このような隠れディレクトリは通常どのサブジェクトに対しても見ることができない。

【００５７】テキスト属性

ＭＡＣラベルの階層要素およびカテゴリー要素を記憶するとき、通常、二進セキュリティ属性と呼ばれる二進数値としてこれらの要素を記憶する。本発明の一態様によ

れば、テキスト属性（５２）は、それらの対応する二進セキュリティ属性（５１）と一緒に記憶される。次いで、あるテキスト属性に関連する二進セキュリティ属性を変更したい場合（例えば、あるラベルの階層要素を「秘密」から「最高機密」に変更する場合）、システムはそのテキスト属性を探索し、それを発見したとき、発見したテキスト属性と共に記憶されている二進セキュリティ属性を変更する。

【００５８】これによって、二進属性を探索しなくてもよいという利点が得られ、該当するテキスト属性には関連のない等価な二進値であってテキスト属性に関連する二進値が変わったからといって単純には変更すべきでない当該等価な二進値の発見が回避できる。このような二進値が変更されるならば、システム上の重要なデータは、どこにあっても失われたり危険に晒される可能性がある。

【００５９】ケーパビリティ・アクセス制御

本発明のセキュリティ・システムは、オブジェクトの所有者に所望量の保護を当該オブジェクトに配置することを許すことによって当該オブジェクトを無許可のアクセスから保護する追加のレベルの制御を提供する。

【００６０】具体的には、オブジェクトの所有者は、オブジェクトに１組のケーパビリティ（要求ケーパビリティ・セット）を割り当てる。プロセスがこのオブジェクトにアクセスするためには（例えば、データ・ファイルへの読出し／書込みアクセスを得るため）、このプロセスは、その有効ケーパビリティ・セットの中に、オブジェクトの所有者がオブジェクトを必要とするケーパビリティ・セットに割り当てたケーパビリティの全てを含まなければならない。

【００６１】ケーパビリティ・アクセス制御（ＣＡＣ）の重要な局面は、このポリシーに対するオーバライドは可能でないことである。即ち、特殊なケーパビリティが、ＣＡＣポリシーをオーバライドするためプロセスにより割り当てられることができ、又は獲得されることができる方法がない。従って、高位の管理者であっても、ＣＡＣ保護オブジェクトへのアクセスを得るためには、必要なケーパビリティ・セット全体を有さなければならない。これにより、管理的乱用に対する高いレベルの保護を与えることになる。

【００６２】セッション・モニタ

セッション・モニタは、ユーザまたは管理者がコンピュータ・システムへのアクセスを初期に得る要領と、ユーザまたは管理者がアクセスのそれらの現在のモードから異なるモードへ（たとえば、ユーザから管理者へ）変更する要領とを制御するセキュリティ・システムの一部である。また、セッション・モニタは、アクセスのそれらの現在のモードが何であるかには拘わらず、ユーザおよび管理者双方のセキュリティに対する信用証明書を前記のユーザまたは管理者のシステムへの初期のアクセス時に確立された最大値に制限することにより、セキュリティ・システムの抑制ポリシーを実施する際にも関係する。

【００６３】セキュリティ・システムの所有者が彼ら自身のソフトウエアを組み込んで、ユーザまたは管理者のシステムへのアクセスを許可する前に、既存モードまたは前記ソフトウエアによって実施される新たなモードのいずれにおいても、ユーザまたは管理者のアクセス・モードを変更したり、認証トランザクションを処理できるという意味において、セッション・モニタは拡張可能に設計されている。いずれの場合でも、供給されるセキュリティ・システムとの一体化は、セキュリティ・システムの一部として供与されるインターフェースに対抗して機能する新たなソフトウエアを書き込み、これもセキュリティ・システムと共に含まれているポリシー・ガイドラインに従うことによって、達成することができる。前記新たなソフトウエアのインストールの後に、元のセキュリティ・システムと共に供与されたアクセス・モードおよび認証方法の管理と同じ機構を用いて、新たなアクセス・モード機能および／または新たな認証機能を同様に達成する。

【００６４】セッション・モニタは、異なるアクセス・モードが異なる信用証明書および異なる認証手順を必要とする場合もあるという概念にも沿うように設計されている。その結果、セキュリティ・システムと共に元々供与される支持されたアクセス・モードか、または所有者によって書き込まれインストールされる支持されたアクセス・モードかのいずれか１つが、前記セキュリティ・システムの所有者によって書き込まれインストールされたソフトウエアを用いて処理される認証ステップを含む、前記アクセス・モードの他のいずれかのものとは異なる一連の認証ステップを必要とする場合も有り得る。同様に、セッション・モニタは、セキュリティ・システムと共に元々供与される支持されたアクセス・モードか、または所有者によって書き込まれインストールされる支持されたアクセス・モードかのいずれかを、異なるセキュリティ信用証明書セットに割り当てることができる。しかしながら、アクセス・モードには拘わらず、いずれのユーザに使用可能なセキュリティ信用証明書は、前記ユーザ（または管理者）のシステムへの初期アクセス時に確立される最大値に限定される。

【００６５】管理セット－許可ケーパビリティ・セット（Administrative Set－Permitted Capability Set）

セキュリティ・システムの一般的な抑制ポリシーのインスタンスの１つは、本発明が管理者のアビリティを制御して、プログラムまたはそのほかのオブジェクトに許可ケーパビリティ・セットを割り当てる方法である。前記管理者が最初に（アクセスが通常のユーザとしてでなければならない）システムにアクセスするとき、セッション・モニタは、前記アクセスのモードにおけるいかなる

変化にも拘わらず、システムへの前記アクセスの期間中にわたって、あらゆるオブジェクトについて許可されたケーパビリティとして、前記ユーザが割り当てることを許可される最大のケーパビリティ・セットを識別する。前記ユーザがそれらのアクセス・モードを変更して管理者になるとき、セッション・モニタはそれらの新たなアクセス・モードの信用証明書においてこの制限を保存する。そして、前記オペレーションを試行するとき、システムは、前記ユーザが現在実行中の役割(role)またはアクセス・モードには関係なく、前記ケーパビリティのみをそのように割り当てることを許可する。

【００６６】従来技術に対する利点は、本発明では、異なるユーザに異なる抑制の限度を割り当てることができ、これは、管理モードも含むそれらの現在のアクセス・モードには関係なく、前記ユーザが試行するあらゆる動作にも適用される。従って、管理者によって許可された動作は、ユーザ毎の細かさで制御可能であり、この特徴を用いて、システム管理に関連する危険性を低減することが可能となる。

【００６７】参照モニタ(Reference Monitor)

参照モニタは、サブジェクトによるオブジェクトへのアクセスの全要求を調停するエンティティであり、従ってサブジェクトにオブジェクトへのアクセスが認可されるか否か、そしてどの程度まで認可されるのかを制御する。このような参照モニタは、上述の従来の技術において論じたデータ・ジェネラルのセキュリティ・システムの初期バージョンにおいて見出すことができる。上述の様々なサブジェクト対オブジェクト・アクセス・ポリシーは、情報セキュリティ・ポリシー表データベースに種々のポリシー・データを記憶することによって実施することができ、これは参照モニタの一部として維持される。

【００６８】情報セキュリティ・ポリシー表データベースは、参照モニタが呼び出してアクセスをチェックするポリシー・モジュールを含む。この表は、システム・ソフトウエアを最初にインストールするときに、特定のコンピュータ・システムの特定のセキュリティ・ポリシーを満足するように構成することが可能である。更に、この表は、企業のセキュリティ・ポリシーが変わるときには、改変することができる。

【００６９】先に論じたデータベースは、オペレーティング・システム・ソフトウエアと共に、ハード・ドライブ、ＣＤ－ＲＯＭまたは半導体に基づくメモリのようなコンピュータ・システムの記憶媒体上に記憶される。コンピュータ・システムは、連続的にこのデータベースにアクセスし、セキュリティ・ポリシーを決定し設定する。

【００７０】本発明は、上述の実施形態によってではなく、特許請求の範囲の精神および範囲によってのみ限定されるものである。

【図面の簡単な説明】

【図１】本発明の好適実施形態による、プログラムを開始するときの、ケーパビリティ・セットおよびそれらの相互作用を示すブロック図。

【図２】本発明の実施形態によるオペレーションのブラケット化を示す図。

【図３】本発明の実施形態によるプロセス上のＭＡＣ範囲を示す図。

【図４】本発明の実施形態によるオブジェクト上のＭＡＣ範囲を示す図。

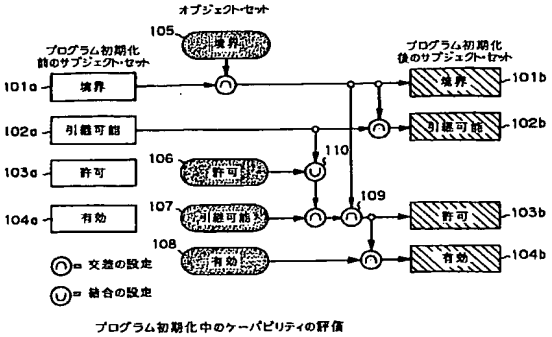
【図５】本発明の実施形態によるＭＡＣ領域を示す図。

【図６】本発明の実施形態によるテキスト属性を示す図。

【符号の説明】

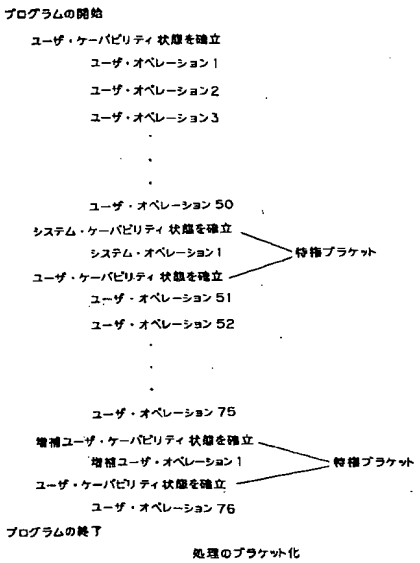
４１　管理領域
４２　ユーザ領域
４３　ウイルス防止領域
５１　二進セキュリティ属性
５２　テキスト属性
１０１ａ，１０１ｂ　サブジェクト境界ケーパビリティ・セット
１０２ａ，１０２ｂ　サブジェクト引継可能ケーパビリティ・セット
１０３ａ，１０３ｂ　サブジェクト許可ケーパビリティ・セット
１０４ａ，１０４ｂ　サブジェクト有効ケーパビリティ・セット
１０５　オブジェクト境界ケーパビリティ・セット
１０６　オブジェクト許可ケーパビリティ・セット
１０７　オブジェクト引継可能ケーパビリティ・セット
１０８　オブジェクト有効ケーパビリティ・セット
１１０　セット結合演算子
１０９　セット交差演算子

【図1】



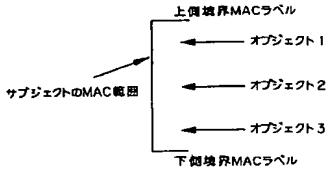
プログラム初期化中のケーパビリティの評価

【図2】



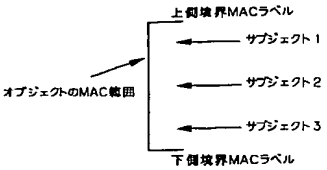
処理のブラケット化

【図3】



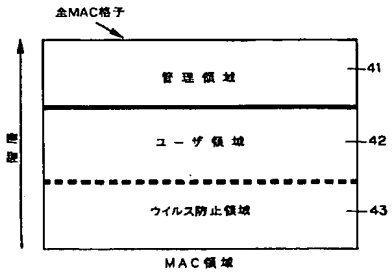
サブジェクトのMAC範囲

【図4】

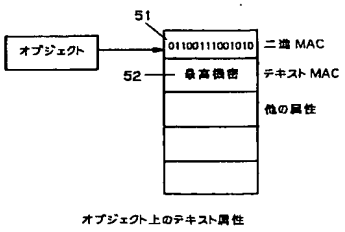


オブジェクトのMAC範囲

【図5】



【図６】



フロントページの続き

(72)発明者　マイケル・ドノヴァン・キーン
アメリカ合衆国フロリダ州33067，コーラル・スプリングス，ノース・ウエスト 8855，ドライブ　47
(72)発明者　エリック・スコット・レウィン
アメリカ合衆国ノース・カロライナ州 27502，アペックス，ペンランド・コート　100
(72)発明者　ウィリアム・ジェームズ・メイヤーズ
アメリカ合衆国ノース・カロライナ州 27709，リサーチ・トライアングル・パーク，ピー・オー・ボックス　12627
(72)発明者　ジョン・フレデリック・スペンサー
アメリカ合衆国ノース・カロライナ州 27609，ローリー，ノースブルック・ドライブ　513
(72)発明者　ミラード・クラムフォード・テイラー・ザ・セカンド
アメリカ合衆国ノース・カロライナ州 27516，チャペル・ヒル，ブライドル・スパー　9306

【外国語明細書】

1. Title of Invention
SECURITY SYSTEM FOR COMPUTER SYSTEMS
2. Claims

1. A security control method for a computer system, said method comprising steps of:

(a) storing in a database of said computer system and in the computer system itself security policies;

(b) continually accessing said database and the computer system during use thereof in order to determine the specific policy stored therein with respect to a particular subject presently using the computer system; and

(c) allowing the subject access to said computer system only to the extent allowed by the results of step (b);

wherein said database and computer system contains Mandatory Access control labels for each subject and each object, said labels being arranged in a hierarchical lattice, with the higher level labels being assigned to an administrative region, the intermediate level labels being assigned to a user region and the lower level labels being assigned to a virus prevention region; and

further wherein, according to the security policy stored in said database and in the computer system, a subject operating in one Mandatory Access Control region is not allowed to access an object in another region unless a special capability or a special set of Mandatory Access Control attributes is assigned to the subject.

2. The method of claim 1 further wherein, according to the security policy stored in said database and embodied in said computer system, subjects are only allowed to run programs which have Mandatory Access Control labels in said virus prevention region.

3. A security control method for a computer system, said method comprising steps of:

(a) storing in a database of said computer system and in the computer system itself security policies;

(b) continually accessing said database and the computer system during use thereof in order to determine the specific policy stored therein with respect to a particular subject presently using the computer system; and

(c) allowing the subject access to said computer system only to the extent allowed by the results of step (b);

wherein said database and computer system contains Mandatory Access control labels for each subject and each object, said labels being arranged in a hierarchical lattice, and

further wherein, according to the security policy stored in said database and the computer system, a range of Mandatory Access Control labels is assigned to a subject and that subject has access to objects having Mandatory Access Control labels within the range.

4. The method of claim 3 wherein according to the security policy stored in said database and computer system, a subject assigned a range of Mandatory Access Control labels is further assigned a set of capabilities which allow it additional accesses to objects having Mandatory Access Control labels within the range.

5. A security control method for a computer system, said method comprising steps of:

(a) storing in a database of said computer system and in the computer system itself security policies;

(b) continually accessing said database and the computer system during use thereof in order to determine the specific policy stored therein with respect to a particular subject presently using the computer system; and

(c) allowing the subject access to said computer system only to the extent allowed by the results of step (b);

wherein said database and the computer system contains Mandatory Access control labels for each

subject and each object, said labels being arranged in a hierarchical lattice, and

further wherein, according to the security policy stored in said database and in the computer system, a range of Mandatory Access Control labels is assigned to an object and that object is accessible to any subject having a Mandatory Access Control label within the range.

6. A security control method for a computer system, said method comprising steps of:

(a) storing in a database of said computer system and in the computer system itself security policies;

(b) continually accessing said database and the computer system during use thereof in order to determine the specific policy stored therein with respect to a particular subject presently using the computer system; and

(c) allowing the subject access to said computer system only to the extent allowed by the results of step (b);

wherein said database and the computer system contains Mandatory Access control labels for each subject and each object, said labels being arranged in a hierarchical lattice with each label having a binary security attribute, and

further wherein said database contains a text attribute for each Mandatory Access Control label, said text attribute being stored alongside the respective binary security attribute for each label.

7. A security control method for a computer system, said method comprising steps of:

(a) storing in a database of said computer system and in the computer system itself security policies;

(b) continually accessing said database and the computer system during use thereof in order to determine the specific policy stored therein with respect to a particular subject presently using the computer system; and

(c) allowing the subject access to said computer system only to the extent allowed by the results of step (b);

wherein said database and the computer system contains a required capability set containing a plurality of capabilities for an object and an effective capability set containing a plurality of capabilities for each subject and

further wherein, according to the security policy stored in said database, a subject is only allowed to access an object if the subject's effective capability set includes all of the capabilities in the object's required capability set.

8. A security control method for a computer system, said method comprising steps of:

(a) storing in a database of said computer system and in the computer system itself security policies;

(b) continually accessing said database and the computer system during use thereof in order to determine the specific policy stored therein with respect to a particular subject presently using the computer system; and

(c) allowing the subject access to said computer system only to the extent allowed by the results of step (b);

wherein said database contains:

a subject inheritable capability set containing a plurality of capabilities which a subject can pass on to future program executions,

a subject bounding capability set containing an absolute upper limit on all capabilities assigned to a subject, and

further wherein, according to the security policy stored in said database and the computer system, a subject is prohibited from passing on capabilities in its subject inheritable capability set when

such capabilities are not included in the subject's bounding capability set.

9. A method of executing a program on a computer, said program including user mode operations which a user specifically requests the computer to perform and which the user usually has the capability to perform, and system mode operations which the system performs for system maintenance or integrity purposes, and augmented user operations in which the system provides a few select capabilities to enhance those provided by the user; said method containing steps of:

setting a subject's effective capability set equal to its inheritable capability set during the time in which a sequential string of user mode operations is being executed by said computer;

setting a subject's effective capability set equal to its permitted capability set during the time in which a string of sequential system mode operations is being executed by said computer; and

setting a subject's effective capability set equal to its inheritable capability set during the time in which a subsequent sequential string of user mode operations is being executed by said computer;

setting a subject's effective capability set equal to its inheritable capability set plus a well-defined set of augmenting capabilities during the time in which a string of sequential augmented user mode operations is being executed by said computer.

10. A security control method for a computer system, said method comprising steps of:

(a) storing in a database of said computer system and in the computer system itself security policies;

(b) continually accessing said database and the computer system during use thereof in order to determine the specific policy stored therein with respect to a particular subject presently using the computer system; and

(c) allowing the subject access to said computer system only to the extent allowed by the results of step (b);

wherein said database contains a plurality of authentication mechanism which a user must correctly respond to before being allowed initial access to the computer system, and

further wherein each user can be assigned a different set of authentication mechanism which the user must correctly respond to before being allowed initial access to the network.

11. A security control method for a computer system, said method comprising steps of:

(a) storing in a database of said computer system and in the computer system itself security policies;

(b) continually accessing said database and the computer system during use thereof in order to determine the specific policy stored therein with respect to a particular subject presently using the computer system; and

(c) allowing the subject access to said computer system only to the extent allowed by the results of step (b);

wherein said database contains an administrative capability set containing a plurality of capabilities, each capability being a different type of privilege which can administrator can confer upon subjects; and

further wherein according to the security policy stored in said database, an administrator is prohibited from assigning a subject a capability which is not in the administrator's administrative capabilities set.

3. Detailed Description of Invention

Field of the invention

The present invention relates to a computer system and specifically to a set of security mechanisms for a computer system. The security system precisely controls who has access to a computer system and the extent of access to the system's resources once the system is accessed.

Background of the invention

Within a single company or government agency, as more and more computers become connected together into large computer networks, sometimes stretching across the globe, it becomes increasingly important to protect the information on the computer system from unauthorized access. Such unauthorized access could come from outside the organization, but oftentimes it comes from within the organization. For example, a lower level employee could gain access to sensitive information concerning employee salaries or future plans of the organization.

Data General Corporation, the assignee of the present application, has been active in this field, and introduced a computer security system in August of 1994. This computer security system was tightly integrated with Data General's industry leading DG/UX Unix-based operating system and formed an integral component of the DG/UX operating system architecture. While this security system provided a certain level of control, it suffered from the following drawbacks.

One of the main problems with the prior security system is that in order to allow a user to perform an administrative action on the system, it was necessary to give this user total access to the entire system. For example, if a lower level administrative user is hired into the administration department of a large company to maintain a log of valid users, this lower level user would have to be given "super-user" access to the entire computer system, and could thus access everyone's private files and could potentially read and/or alter company sensitive data. Another problem involves viruses that can be introduced in a computer system. The prior system did not isolate viruses to

protect valuable files, nor did it take positive steps to prevent viruses from occurring in the first place, even if being run by an administrative user.

Summary of the invention

The present invention is drawn to a security system for a computer system in which specific limitations are imposed on who has access to exactly what computer functions and data on the computer system. As a consequence of the specific implementation of the security system of the present invention, viruses are securely contained and prevented from expanding into areas where they can destroy stored programs or data. Viruses are also prevented from being introduced or propagating in a large number of instances.

The security system of the invention basically involves breaking up the totality of computer functions into required privileges and assigning different privileges to each user depending on the particular job which that user is to do on the computer system.

Also, security labels are placed on each data file or other system resource, and on each user process. A hierarchy of labels is created ranging from highly secret to commonly accessible and strict policies are enforced by the security system based on these labels to determine who has what type of access to which data files or other system resource. According to the invention, a range of these labels is assigned to a particular user process to define a clearance range in which the process is allowed to operate. Further, the hierarchy of labels is divided into a small number (for example 3) of regions, and a user process operating on one region is generally not allowed to access another region except in a very carefully proscribed manner.

Further, an owner of a data file is allowed to place restrictions on the file so that only users who posses certain capabilities can gain access to the file.

Detailed Description of the preferred embodiments

Definitions

The following commonly occurring terms in this application are hereby defined as follows:

subject: an active entity of the computer system which either accesses objects or causes the system state to change (for example, a process operating on a user's behalf).

process: an actively running program, often running on a user's behalf with a set of credentials identifying who that user is and embodied with the access rights of that user.

object: a target of an operation, often a passive element of the computer system which receives and stores data (for example, a data file or a program).

operation: a request for service from the system (for example, a request to open a file for read and write access).

Capability mechanism for privileges

The security system of the present invention allows each process to be assigned a set of capabilities (or privileges) which it can invoke during its existence. One example of a capability is the ability to open a file to which the system would normally deny access, another is performing a

restricted operation, such as shutting down the system, which is not available to regular users. This set of possible capabilities which a process may invoke when needed is called the subject permitted capability set (see 103a and 103b in Figure 1).

Each process also has a subject effective capability set (104a and 104b) which is a subset of the permitted capability set, and is the set of capabilities that the process is currently asserting to perform operations. That is, the effective set is what the system considers when making an access control decision, while the permitted set is the upper limit of what can be included in the effective set.

Sometimes during execution of a particular program, a process' permitted capability set must be temporarily expanded to include capabilities which the user does not possess in the normal course of business. For example, the user may decide to change his password. To do this, the user must update the system files which store his password and other associated information. However, the user is not usually allowed access to these files, so the change password program must provide the ability to access these files. The program is responsible for ensuring that the extra capabilities are only used for the purpose for which they were intended, and the extra capabilities are automatically removed when the change password program terminates.

Such extra capabilities are stored as the object permitted capability set (106) on the program. The capabilities in this set are temporarily granted to a process while it is running that program. The object may also have other capability sets associated with it, which change the capabilities assigned to any process that runs the program as shown in Figure 1. For example, when a process starts running a program, the object permitted capabilities are added to the process' permitted

capabilities by a set union operator (110), and the subject permitted set is limited to the subject bounding set (discussed below) by the set intersection operator (109).

The set 102a is the subject inheritable capability set and represents the set of capabilities which a process can pass on to later programs run by the process. That is, the capabilities in this set are the ones that this process normally possesses, without additions from any object permitted capability set. The subject inheritable set before program initialization (102a) basically becomes the new subject inheritable set after the program starts (102b), except for Applicant's added security measures concerning a bounding set, which will be explained below. It is by this mechanism that capabilities acquired from an object permitted capability set are prevented from passing to later programs, since such enhanced capabilities are never added to the subject inheritable set.

The above structure has been included in IEEE POSIX security system in the past. Applicant has added a failsafe mechanism to the above basic structure as follows.

A subject bounding capability set (101a, 101b) is defined, and is the largest group of capabilities which may ever be acquired by a process. It is not possible for a process to increase its bounding set. As shown in Figure 1, this set may be decreased (for example if an object bounding set is smaller than the original subject bounding set), and it serves as a limit on all other subject capability sets. This effectively reduces the process' ability to acquire capabilities and thus serves as a failsafe mechanism. For example, a user who is not authorized to perform administrative functions may be assigned a relatively small bounding set. Even if this user manages to take on an administrative identity, their bounding set will prevent them from acquiring the associated capabilities, and thus greatly reduce (or even eliminate) the potential for damage to the system.

As a further check, each executable program may also have an object bounding capability set (105) which is intersected with the subject bounding capability set (101a) during initialization of a program to form the new subject bounding set (101b) which will be effective while the program runs. Since this new bounding set also limits all the other subject capability sets, in particular the permitted and effective sets (103b and 104b respectively), this provides a mechanism for preventing a given program from being run with particular capabilities.

Event Mechanism

If the implementation were to directly check for the presence of a particular capability in the process' subject effective set whenever it wished to check for privilege, the actual security policy being enforced with respect to capabilities would have to be determined by Applicant, and individual sites would have to use that same policy. To avoid this, and allow the security policy enforced by the system to be configurable by each site to meet its individual requirements, privilege checks are done by means of "events". An event is defined as "a place in the code where a security-relevant decision is made or recorded". When the system needs to check whether a process should be allowed to perform some operation, it uses the unique event name defined for that particular privilege check to look up, in a table, the capability or capabilities which the process must possess to pass the check. By providing a mechanism for individual sites to modify the event name-to-capability table, the site can configure the system to enforce their own security policy, in addition to, or instead of, the supplied event/capability policy delivered with the security system.

Operation Bracketing

While a subject is executing an object program, the state of the process' subject effective capabil-

ity set (104b) is changed periodically to reflect the type of operation currently being performed. Three different types of operations are defined.

Firstly, a *user operation* is one that should be performed with only the capabilities that are normally assigned to the user. For example, when a user requests a file be printed, accessing the requested file is a user operation. In this case, the process' effective capability set (104b) is set equal to its inheritable set (102b), since this contains the user's capabilities with no contribution from an object permitted set (106). This state is maintained as long as the program performs user operations.

Secondly, a *system operation* is one which the system needs to succeed if possible. For example, in the print request above, the system may copy the file to a spool area, which is normally inaccessible to users, before printing it. This copy is a system operation, since it must succeed for the file to print. In this case, the process' effective capability set (104b) is set equal to its permitted capability set (103b), which represents the user's capabilities plus any object permitted capabilities (106).

Finally, an *augmented user operation* is one which needs access to one or more capabilities from the object permitted set (106), but despite this is essentially a user operation. An example is setting an object permitted capability set on a program, which requires a special capability that no user or administrator normally possesses. If the program to set such a capability set decides to allow the operation, it needs to enable this particular capability, but it should not enable other object permitted capabilities it may have, since this may grant the user access to the target program which he would not normally have. In this case, the process' effective set (104b) is set equal

to the subject inheritable set (102b) with the addition of particular capabilities from the subject permitted set (103b).

The term "bracketing" refers to enabling extra capabilities in the subject effective set (104b) just before a system or augmented user operation and disabling them immediately after the operation. In the prior art, bracketing was done around each user operation (even though there may have been many in a row) This involved significant analysis, to determine the capabilities required for each operation, and also computational effort, to compute and modify the effective set (104b) so often.

According to the present invention, as shown in Figure 2, the program establishes the effective capability state (104b) for user operations when it starts. This capability state is maintained as long as the program is performing user operations. When it comes time to perform a system or augmented user operation, the appropriate capabilities are enabled in the effective capability set (104b) immediately prior to the operation, and the user-operation capability set is reset in the effective capability set immediately after the operation. This enabling and disabling of extra capabilities before and after the system or augmented user operation form a "bracket" around the operation. In this way, the program is usually running with only the user's capabilities, except at specific places where we identify system or augmented user operations, when extra capabilities are enabled. Enabling all available capabilities for a system operation instead of just those specifically needed for that operation is acceptable, since all operations are such that having extra, unrelated, capabilities will not affect the outcome of the operation.

Since the present invention brackets much less often than the prior art, much analysis and compu-

tational effort is saved. That is, there is no need to adjust the effective capability set (104b) for each user operation. Instead, the capabilities with which the user normally runs are used.

Mandatory Access Control

The above discussion of capabilities related to the security system's ability to allow checks to be placed on a process concerning exactly which operations it can perform. According to the invention, further checks can be placed on a subject's access to particular objects.

According to well-known Mandatory Access Control (MAC) theory, labels are applied to each object and each subject. Each label consists of two components:

1. a hierarchy component, which is ordered and denotes a level of sensitivity (e.g., unclassified, secret, top secret, etc.); and
2. a category set component, which is non-ordered and denotes the area(s) of interest (e.g., marketing, research-and-development, personnel, etc.).

These MAC labels form a mathematical lattice. An example of such a label is "Secret:(SuperCar, Marketing)". This label refers to an object (e.g., a data file containing market research information) which is at the "Secret" level of sensitivity and relates to both the "SuperCar" and "Marketing" areas of interest.

A first label *dominates* (i.e., is higher than) a second label if

1. the hierarchical component of the first label is greater than or equal to the hierarchical component of the second label and

2. the category set of the second label is a subset (i.e., is contained within) the category set of the first label.

Thus, given two labels, the first may dominate the second, the second may dominate the first, each may dominate the other (in which case the two labels are equal), or neither may dominate the other (in which case the labels are said to be "incomparable").

A subject can read an object when the subject's label dominates the object's label (including the case where the labels are equal). This is call "read-down". A subject can write to an object when the object label dominates the subject label, which is called "write-up".

The present invention implements a stricter base policy of "read-down, write-equal", where a subject can write to an object only if the subject and object labels are equal

MAC ranges

In the prior art, a process running at a particular MAC label could only read an object with a label dominated by the process' label. If it were desired to give a user more access than this base policy allowed, the user's process would had to be granted total access to all objects in the computer system. In the UNIX operating system, such a user is known as a "superuser" (or "root"). In order to perform one restricted operation, a user had to be given the ability to perform all restricted operations. This is typically highly undesirable in practice.

Applicant has overcome this problem by a combination of the capability mechanism described above and assigning a MAC range (a range of MAC labels) to a process (in addition to the process' MAC label). As shown in Figure 3, the range has an upper bound (the highest MAC label

in the range) and a lower bound (the lowest MAC label in the range), and the upper bound dominates the lower bound. Such a MAC range forms a sublattice of the entire MAC lattice.

The capability mechanism provides capabilities which allow a process to override the basic system MAC policy when accessing objects with labels within the process' MAC range. For example, a process' capabilities can be configured to allow write access to any object whose MAC label is higher than the process' label but still within the process' MAC range ("write-up-within-range"). Other capabilities allow "read-up-within-range" and "write-down-within-range", which have definitions similar to the above.

Applicant's MAC ranges, combined with the capability mechanism, thus allows a user to be given increased access to a well-defined set of objects (i.e., those with MAC labels within the process' MAC range) without giving that user any increased access to other objects on the system.

In addition to the above embodiment (see Figure 4), a MAC range can be defined with respect to a particular object. As long as the subject's MAC label is in between the upper and lower bounds of the object's MAC range (each of subjects 1, 2 and 3 in Fig. 4), it is granted write-only access to the object. Read access requires that the subject's MAC label dominate the upper bound of the object's MAC range. Read/write access is thus only granted to a process whose MAC label is equal to the upper bound of the object's range.

This way, a range of users at various MAC labels can be allowed to write to an object (e.g., a log file in which users record daily transactions) without necessarily giving them the ability to read what others have written.

MAC regions

The basic MAC policy (read-down, write-equal) has been found by Applicant to be limiting and to result in various problems.

Firstly, if an administrator is performing an administrative action, he/she is able to "read down" to any MAC label below the label of their process. Many administrative processes are typically run in the upper portion of the MAC lattice, so that regular (non-administrative) users cannot access administrative objects. Thus if a low-level user (e.g., a high-school student) is given a limited administrative role (such as adding and removing users in a large company which gains and loses many employees each month), that user would be able to access anything dominated by their administrative MAC label. As an example, the low-level administrative employee could gain access to the electronic mailboxes of the senior company officials.

Secondly, a program run by a user may contain malicious code, such as a virus, which could modify or destroy other programs and files on the system. If an administrator runs such a program, the system executables may be modified, and may perform unauthorized actions when later run by other users, such as disclosing information intended to be kept secret.

Applicant has solved these problems by dividing the totality of MAC hierarchies into a small number of distinct and non-overlapping regions. For example, as shown in Figure 5, three regions can be used: an administrative region (41), a user region (42) and a virus prevention region (43). The administrative region is located at the highest MAC hierarchies, the virus prevention region is at the lowest MAC hierarchies, and the user region is in between. Effective policy restrictions are placed on processes in one region accessing objects in another region so that problems such as

described above are overcome.

For example, a process running with a MAC label in the administrative region gains access to objects in the user and virus prevention regions not though its (administrative) MAC label but rather through the upper bound of its MAC range (which by definition is in the user region). By giving an administrator a MAC range whose upper limit is at the bottom of the user region, it is thus possible to prevent the administrator from accessing most or even all of the files in the user region.

Likewise, a special capability is required to write to the virus prevention region, and this capability is typically granted only to the administrator responsible for installing software, not to other administrators or users. The system executables are stored in the virus prevention region so they gain the benefit of this write-protection. When a process runs a program, a copy of the program is made into memory allocated to that process. If the program contains malicious code which attempts to modify any objects in the virus prevention region (where all system executables and many other system files are placed), access will be denied. The virus is thus effectively contained and is unable to spread and do further damage.

There are no capabilities which allow a process to override the restrictions on access to objects in different MAC regions. This allows for the containment of administrative authority by limiting the access of various administrators to only those objects they need to get their job done.

Trusted Facility Mode

Further to the MAC regions aspects discussed above, the present invention provides extra security

and extra virus protection in that a process is only allowed to execute a program if that program is stored in the virus prevention region (43). This can be achieved, for example, by performing a check on each program before it is executed to make sure the software resides in the virus prevention region (43). The virus prevention region (43) is thus the only place which can contain potentially executable programs for processes running in Trusted Facility Mode. This provides an expanded amount of control because the system administrators can decide exactly which programs will be made available to users.

With the Trusted Facility Mode aspect of the invention, users are unable to, for example, download and run programs from the Internet (using e.g., File Transfer Protocol). External programs are a common source of viruses. Therefore, by preventing external programs from being run on the computer system, viruses can be prevented in a large number of instances.

Trusted Facility Mode also helps to prevent administrative abuse by preventing administrators from running programs that are not officially installed in the virus prevention region.

Multilevel Directories

When creating a file system object, a subject is actually writing to a directory listing each file sys-

subject carries a special capability designed for this purpose.

Text Attributes

When the hierarchical and categorical components of MAC labels are stored, they are generally stored as binary values called binary security attributes. According to an aspect of the invention, text attributes (52) are stored alongside their corresponding binary security attributes (51). Then, if it is desired to change the binary security attributes associated with a text attribute (e.g., if the hierarchical component of a label is to be changed from "secret" to "top secret") the system searches for the text attributes and when it finds it, the binary security attribute stored alongside the found text attribute is changed.

This creates the advantage of not having to search for the binary attribute, which avoids finding equivalent binary values that are not associated with the text attribute and should not change simply because the binary value associated with the text attribute has changed. If such binary values were changed, important data elsewhere on the system could be lost or compromised.

Capability Access Control

The security system of the invention provides an additional level of control in which the owner of an object is allowed to place a desired amount of protection on the object, to protect the object from unauthorized access.

Specifically, the owner of an object assigns a set of capabilities (a required capability set) to an object. In order for a process to access the object (e.g., to obtain read/write access to a data file), the process must contain, in its effective capability set, all of the capabilities which the object owner has assigned to the object required capability set.

An important aspect of Capability Access Control (CAC) is that no overrides to this policy are possible. That is, there is no way that special capabilities can be assigned to or acquired by an process to override the CAC policy. Therefore, even high level administrators must have the entire required

capability set to gain access to a CAC protected object. This provides a high level of protection against administrative abuse.

Session Monitor

The Session Monitor is the part of the security system which controls the manner in which a user or administrator initially gains access to the computer system, and the manner in which a user or administrator changes from their current mode of access to a different mode (for example, from user to administrator). The Session Monitor also participates in enforcing the security system's containment policy by limiting both user and administrator security credentials, no matter what their current mode of access, to maximum values established upon the said user's or administrator's initial access to the system.

The Session Monitor has been designed to be extensible, in the sense that the owner of the security system can incorporate their own software to change access mode of a user or administrator, or to process authentication transactions before allowing a user or administrator access to the system, in either an existing mode or a new mode implemented by the said software. In either case, integration with the supplied security system is accomplished by writing the new software to function against interfaces delivered as part of the security system, following policy guidelines also included with the security system. After installation of said new software, administration of the new access mode function(s) and/or the new authentication function(s) is accomplished in the same way, using the same mechanisms, as administration of the access modes and authentication method(s) delivered with the original security system.

The Session Monitor has also been designed around the concept that different access modes might

require different credentials and different authentication procedures. Consequently any one of the supported access modes, either originally delivered with the security system or written and installed by the owner, may require a different series of authentication steps than any other of the said access modes, including authentication steps that are processed using software written and installed by the owner of the said security system. Similarly, the Session Monitor may assign any supported access mode, either originally delivered with the security system or written and installed by the owner, a different set of security credentials. However, the security credentials available to any user, regardless of access mode, are limited to maximum values established upon the said user's (or administrator's) initial access to the system.

Administrative Set-Permitted Capability Set

One instance of the security system's general containment policy is the manner in which the invention controls an administrator's ability to assign permitted capability sets to programs and other objects. When the said administrator first accesses the system (which access must be as an ordinary user), the Session Monitor identifies a maximum set of capabilities that the said user will be allowed to assign as permitted capabilities on any object throughout the term of the said access to the system, notwithstanding any changes in the mode of said access. When the said user changes their access mode to become an administrator, the Session Monitor preserves this limit in the credentials of their new access mode; and when the said operation is attempted, the system permits only the said capabilities to be so assigned, without regard to the role or access mode in which the said user is currently executing.

The advantage over prior art is that in the present invention different users may be assigned differ-

ent containment limits, which apply to any action the said users attempt without regard to their current access mode, including administrative modes. Thus the actions allowed by administrators can be controlled to a per-user granularity; which feature can be used to reduce the risks associated with system administration.

Reference Monitor

The Reference Monitor is the entity that mediates all requests for access to an object by a subject, and thus controls whether, and to what extent, the subject is granted access to the object. Such a Reference Monitor can be found in the earlier version of Data General's security system discussed in the Background of the Invention above. The various subject-to-object access policies described above can be implemented by storing various policy data in an Information Security Policy Table database, which is maintained as part of the Reference Monitor.

The Information Security Policy Table database contains policy modules which the Reference Monitor must invoke to check access. This table can be configured when the system software is first installed, to meet the specific security policy of the specific computer system. Further, the table can be altered when the security policy of the enterprise changes.

The database discussed above is stored, along with the operating system software, on a computer system storage medium such as a hard drive, CD-ROM or semi-conductor based memory. The computer system continually accesses this database to determine and set the security policy.

The present invention is not limited by the above-described embodiments, but only by the spirit and scope of the appended claims.

4. Brief Description of Drawings

Figure 1 shows a block diagram of capability sets and their interaction when starting a program according to a preferred embodiment of the present invention;

Figure 2 illustrates operation bracketing according to an embodiment of the present invention;

Figures 3 and 4 illustrate MAC ranges on process and objects respectively, according to embodiments of the present invention;

Figure 5 illustrates MAC regions according to an embodiment of the present invention; and

Figure 6 illustrates Text Attributes according to an embodiment of the present invention.

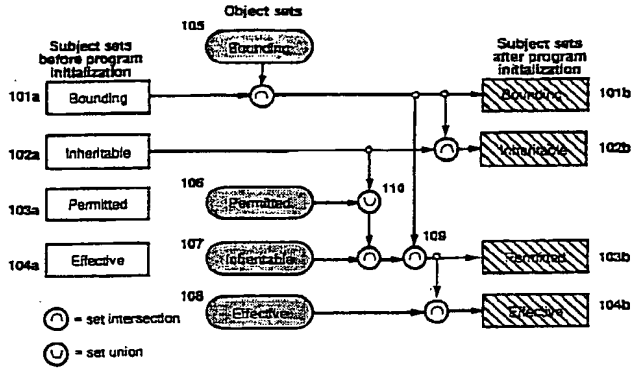


Figure 1: Evaluation of Capabilities during Program Initialization

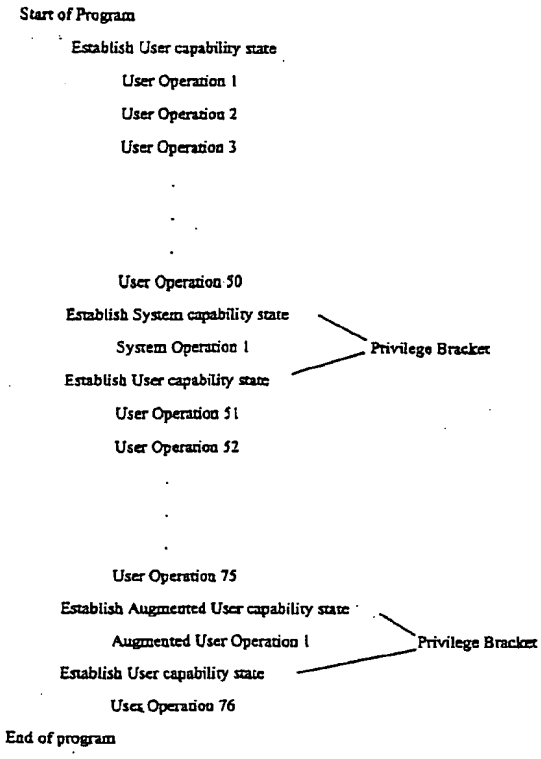


Figure 2: Operation Bracketing

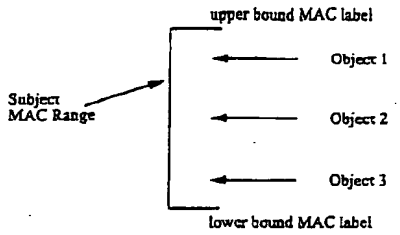


Figure 3 : Subject MAC Range

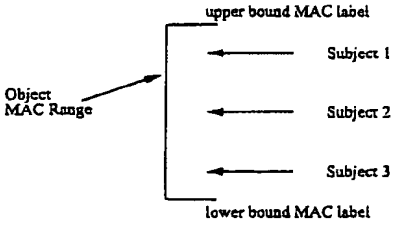


Figure4 : Object MAC Range

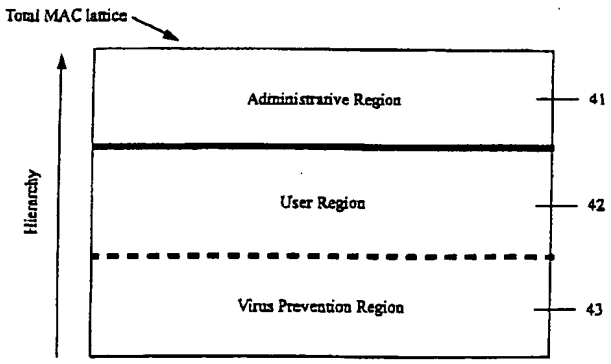


Figure5: MAC Regions

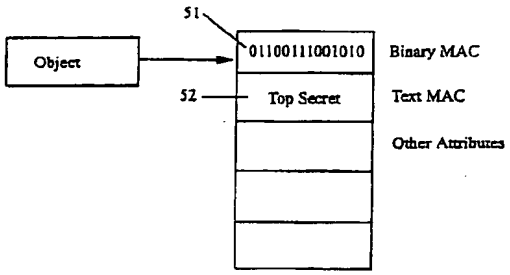


Figure 6 Text Attributes on Objects

1. Abstract

A security system for a computer system imposes specific limitations on who has access to the computer system and to exactly what operations and data. Viruses are securely contained and prevented from expanding into areas where they can destroy stored programs or data. Viruses are also prevented from being introduced or executed in a large number of instances. The totality of computer functions is broken up into a set of events with an associated set of capabilities and different capabilities are assigned to each user depending on the particular job which that user is to do on the computer system. Also, security labels are placed on each data file and other system resources, and on each process. Further, a range of hierarchy/category labels (MAC labels) is assigned to each process to define a sub-lattice in which special capabilities can apply. Further, the hierarchy of labels is divided into a small number (for example 3) of regions, and a process operating in one region is generally not allowed to cross over into another region. Further, an owner of a data file is allowed to place restrictions on the file so that only users who posses certain privileges can gain access to the file.

2. Representative Drawing
Fig.1